# Diseño Conceptual de un Datawarehouse Temporal en el Contexto de MDA

**Carlos G. Neil**
Universidad Abierta Interamericana
Facultad de Tecnología Informática
Buenos Aires, Argentina
Carlos.Neil@vaneduc.edu.ar

y

**Claudia F. Pons**
Universidad Abierta Interamericana
Facultad de Tecnología Informática
Buenos Aires, Argentina
cpons@info.unlp.edu.ar

## Abstract

Model-Driven Architecture (MDA) is becoming a widely accepted approach for developing complex software systems. MDA advocates the use of models as the key artifacts in all phases of development, from system specification and analysis, to implementation. Model transformations are the engine of MDA, they are performed starting from a platform independent model with the aim of making the model more platform-specific at each step. Adhering to the MDA philosophy, in this article we present a methodology for the conceptual design of a temporal datawarehouse, which allow us to define the concepts in an implementation-independent way. Our proposal consists in the definition of metamodels and formal transformation rules that provide a framework for the refinement of a temporal data model in order to obtain a temporal multidimensional model by applying the MDA approach.

**Keywords:** MDA, Metamodels, Temporal DW, Model Transformations

## Resumen

Model-Driven Architecture (MDA) se está convirtiendo en un enfoque ampliamente aceptado para el desarrollo de sistemas de software complejos. MDA propone el uso de modelos como artefacto clave en todas las fases de desarrollo, desde la especificación y análisis hasta la implementación. La transformación de modelos es la base de MDA, comenzando por un modelo independiente de la plataforma el objetivo es lograr, en cada paso, modelos más específicos. Adhiriendo a la filosofía MDA, en este artículo presentamos una metodología para el diseño de un datawarehouse temporal que permite definir los conceptos de manera independiente de la implementación. Nuestro propósito consiste en la definición de metamodelos y reglas de transformación formales que provean un marco para el refinamiento de un modelo de datos temporal para la obtención de un modelo multidimensional temporal aplicando el enfoque MDA
.

**Palabras Claves:** MDA, Metamodelos, DW temporal, Transformación de Modelos

# Diseño Conceptual de un Datawarehouse Temporal en el Contexto de MDA

**Carlos G. Neil**
Universidad Abierta Interamericana
Facultad de Tecnología Informática
Buenos Aires, Argentina
Carlos.Neil@vaneduc.edu.ar

y

**Claudia F. Pons**
Universidad Abierta Interamericana
Facultad de Tecnología Informática
Buenos Aires, Argentina
cpons@info.unlp.edu.ar

**Abstract**

Model-Driven Architecture (MDA) is becoming a widely accepted approach for developing complex software systems. MDA advocates the use of models as the key artifacts in all phases of development, from system specification and analysis, to implementation. Model transformations are the engine of MDA, they are performed starting from a platform independent model with the aim of making the model more platform-specific at each step. Adhering to the MDA philosophy, in this article we present a methodology for the conceptual design of a temporal datawarehouse, which allow us to define the concepts in an implementation-independent way. Our proposal consists in the definition of metamodels and formal transformation rules that provide a framework for the refinement of a temporal data model in order to obtain a temporal multidimensional model by applying the MDA approach.

**Keywords:** MDA, Metamodels, Temporal DW, Model Transformations

**Resumen**

Model-Driven Architecture (MDA) se está convirtiendo en un enfoque ampliamente aceptado para el desarrollo de sistemas de software complejos. MDA propone el uso de modelos como artefacto clave en todas las fases de desarrollo, desde la especificación y análisis hasta la implementación. La transformación de modelos es la base de MDA, comenzando por un modelo independiente de la plataforma el objetivo es lograr, en cada paso, modelos más específicos. Adhiriendo a la filosofía MDA, en este artículo presentamos una metodología para el diseño de un datawarehouse temporal que permite definir los conceptos de manera independiente de la implementación. Nuestro propósito consiste en la definición de metamodelos y reglas de transformación formales que provean un marco para el refinamiento de un modelo de datos temporal para la obtención de un modelo multidimensional temporal aplicando el enfoque MDA
.

**Palabras Claves:** MDA, Metamodelos, DW temporal, Transformación de Modelos

# 1 INTRODUCCIÓN

Las empresas utilizan los datos acumulados durante años, empleados en las transacciones comerciales, para ayudar a comprender y dirigir sus negocios; con ese propósito, los datos de las diferentes actividades se almacenan y consolidan en una base de datos central denominada datawarehouse. Los analistas lo utilizan para extraer información de sus negocios que les permita tomar mejores decisiones [12]. Un datawarehouse es una colección de datos no volátiles, que varían en el tiempo, que están orientados a un tema determinado y que se utilizan para tomar decisiones organizacionales [8]. El modelo multidimensional constituye la base del datawarehouse, en él la información se estructura en hechos y dimensiones; un hecho es un tema de interés para la empresa, se describe mediante atributos denominados medidas o atributos de hecho, los cuales están contenidos en celdas o puntos en el cubo de datos. Un cubo de datos es una representación multidimensional de datos que pueden verse desde distintos puntos de vista; está formado por dimensiones que determinan la granularidad para la representación de hechos y jerarquías que muestran cómo las instancias de hechos pueden ser agrupadas y seleccionadas para los procesos de toma de decisión [3].

En el datawarehouse el tiempo es una de las dimensiones para el análisis [8], [9] pero éste hace referencia al momento en que se realizó una transacción, no se detalla cómo y cuándo varían los atributos o interrelaciones involucradas en esas transacciones. La necesidad de registrar valores que permitan evaluar tendencias, variaciones, máximos y mínimos, justifican considerar en el diseño del datawarehouse cómo algunos atributos o interrelaciones pueden variar en el tiempo. Un esquema multidimensional temporal que incluya, además del hecho principal de análisis en el datawarehouse, esquemas temporales (que no pertenezcan a la jerarquía dentro de las dimensiones) registrará la variación de los atributos o interrelaciones que se modifiquen en el tiempo.

Para la construcción del esquema temporal [20], se adaptó un algoritmo que permitió en forma semiautomática construir, a partir de un modelo entidad interrelación, el diseño conceptual de un datawarehouse [9]. Se utilizó una extensión del modelo entidad interrelación ampliándolo con atributos, entidades e interrelaciones temporales y, aplicando un algoritmo recursivo, se construyó el esquema conceptual, unificando en un sólo modelo, tanto el esquema multidimensional como el temporal; esto permitió registrar y analizar las variaciones temporales así como la realización de consultas sobre la estructura multidimensional.

La transformación, de un modelo de datos temporal a un modelo multidimensional temporal, se realizó de manera informal en dos etapas: primero, utilizando un algoritmo recursivo, se creó un grafo de atributos; luego, utilizando el grafo de atributos más un conjunto de decisiones de diseño para la determinación de cuáles serán dimensiones, jerarquías y medidas, se derivó el modelo multidimensional temporal. La realización de estos pasos, si bien están detallados en la metodología de diseño propuesta, no están formalizados de manera conveniente y necesaria para poder ser automatizados.

La construcción de software, en particular la construcción de un datawarehouse, se enfrenta a continuos cambios en las tecnologías de implementación (OLAP relacional, OLAP Multidimensional, OLAP Híbrido) lo que implica realizar grandes esfuerzos en el diseño de la aplicación, para integrar las diversas tecnologías; también en el mantenimiento, para adaptar la aplicación a los cambios en los requisitos, como a los cambios en las tecnologías de implementación.

Model Driven Architecture [17] fue establecida como una arquitectura para el desarrollo de aplicaciones que tiene como objetivo proporcionar una solución para los cambios de negocio y de tecnología, permitiendo construir aplicaciones independientes de la implementación; representa un nuevo paradigma en donde se utilizan modelos del sistema, a distinto nivel de abstracción, para guiar todo el proceso de desarrollo. La idea clave subyacente es que, si se trabaja con modelos, se

obtendrán importantes beneficios tanto en productividad, portabilidad, interoperatividad y mantenimiento. Podemos dividir el proceso MDA en tres fases; en la primera se construye un modelo independiente de la plataforma (PIM), éste es un modelo de alto nivel del sistema independiente de cualquier tecnología; luego, se transforma el modelo anterior a uno o varios modelos específicos de la plataforma (PSM), éste modelo es de más bajo nivel que el PIM y describe al sistema de acuerdo con una tecnología de implementación determinada; por último, se genera el código fuente a partir de cada PSM. La división entre PIM y PSM está vinculado al concepto de plataforma, el cual no está bien definido, razón por la cual la línea divisoria entre PIM y PSM no está claramente delimitada.

MDA también presenta un modelo independiente de los aspectos computacionales (CIM) que describe al sistema dentro de su ambiente y muestra lo que se espera de él sin exhibir detalles de cómo será construido. El beneficio principal del enfoque MDA es que una vez que se ha desarrollado cada PIM podemos derivar automáticamente el resto de los modelos aplicando las correspondientes transformaciones en forma vertical. Sin embargo, pueden aplicarse también transformaciones horizontales; esto es, un modelo fuente se transformará en un modelo destino dentro del mismo nivel de abstracción [19]. La transformación de PIM a PIM se utiliza cuando los modelos son ampliados, filtrados o especializados durante el desarrollo del ciclo de vida sin necesidad de información dependiente de la plataforma. Una de las más obvias transformaciones es la realizada entre el análisis y el diseño, concepto relacionado con el refinamiento de modelos [17].

En MDA se describen tres opciones para la transformación de modelos: que ésta sea totalmente manual y sea realizada por los desarrolladores; que sea necesaria la participación de desarrolladores en la transformación pero éstos estén guiados por herramientas automatizadas o, por último, que sean generadas automáticamente a partir de los modelos y sin intervención humana.

Aplicando los conceptos de MDA en la construcción de un datawarehouse, identificamos un CIM que define los requerimientos desde un punto de vista de un datawarehouse en su ambiente de negocio; un PIM que lo define desde un punto de vista conceptual sin tener en cuenta ningún detalle tecnológico específico y uno o más PSM´s que especifican aspectos de diseño en distintas plataformas, por ejemplo, ROLAP (OLAP Relacional), MOLAP (OLAP Multidimensional) u HOLAP (OLAP Hibrido) [16].

En el presente trabajo proponemos, dentro del marco de la filosofía MDA, formalizar la transformación del algoritmo presentado; en este caso, proponemos una transformación horizontal (de PIM a PIM), desde el modelo de datos temporal (MDT) al modelo multidimensional temporal (MMT) pasando por un grafo de atributos; utilizaremos un metamodelo para cada uno de los modelos propuestos y aplicaremos sentencias OCL (Object Constraint Language) [22], para formalizar las transformaciones.

El resto del trabajo está estructurado de la siguiente forma: en el capítulo 2 presentamos la transformación informal, tanto del modelo de datos al grafo de atributos, como la transformación del grafo de atributos al modelo multidimensional; en el capítulo 3 mostramos los metamodelos de datos, de grafos y el multidimensional y las dos transformaciones formales; en el capítulo 4 detallamos los trabajos relacionados, tanto al diseño conceptual de datawarehouse temporal, como a las propuestas de diseño de un datawarehouse en un ambiente MDA; por último, en el capitulo 5 presentamos la conclusión y los trabajos futuros.

## 2 TRANSFORMACIÓN INFORMAL DEL MODELO DE DATOS

La metodología de transformación del modelo de datos temporal al modelo multidimensional temporal plantea una serie de pasos, descritos de manera informal, que detallaremos a continuación pero que, en resumen, consisten en la aplicación de un algoritmo que tiene como entrada un modelo entidad interrelación temporal y como salida un modelo multidimensional temporal. Presentamos

con un ejemplo (Figura 1) cómo, utilizando el algoritmo, transformamos un modelo de datos temporales a un grafo de atributos y luego, a partir de éste, creamos el modelo multidimensional temporal. Para la aplicación del algoritmo recursivo transformamos el diagrama entidad interrelación (Figura 1, lado izquierdo) a un diagrama entidad interrelación temporal[1] (Figura 1, lado derecho). El atributo multivaluado se convertirá en una entidad débil con una interrelación temporal (marcada con T) y la interrelación temporal se transformará en una entidad con interrelaciones binarias (marcada con T) vinculadas a las entidades participantes [20]. En los casos en donde sea conveniente preservar una futura jerarquía, proponemos mantener las dos interrelaciones (la instantánea y la temporal). En el ejemplo (Figura 1, lado derecho), conservamos la interrelación entre *PROVEEDOR* y *LOCALIDAD.*
Con el mismo criterio general utilizado para transformar interrelaciones temporales, transformamos la interrelación *venta* en una entidad *VENTA* (Figura 1, lado derecho).

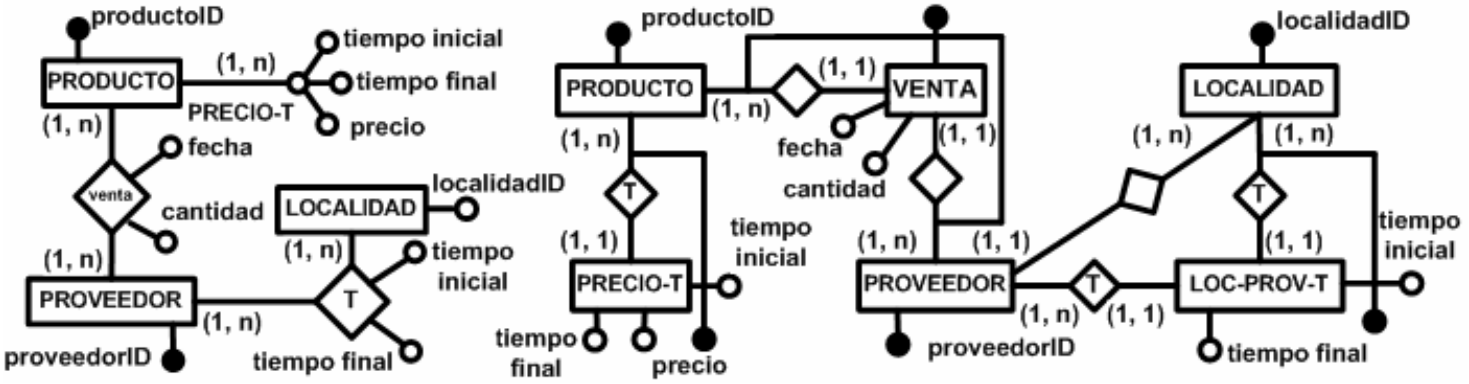


Figura 1. Modelo de Datos y Modelo de Datos Transformado

## 2.1 Transformación Informal del Modelo de Datos Temporal al Esquema de Hecho

Dado un modelo entidad interrelación temporal, se derivará el esquema conceptual de un datawarehouse que contemplará en su diseño aspectos temporales.
Los hechos, como conceptos de interés primario para el proceso de la toma de decisión, corresponden a sucesos que ocurren dinámicamente en la realidad, éstos pueden ser representados en el modelo entidad interrelación temporal mediante una entidad E o por medio de una interrelación R n-aria entre entidades $E_1 ... E_n$. [9].
Dada un área de interés en un modelo entidad interrelación temporal y una entidad E que pertenece a él, denominamos grafo de atributos al grafo tal que:

- Cada vértice corresponde a un atributo, simple o compuesto del modelo entidad interrelación.
- La raíz corresponde al identificador de E.
- El atributo correspondiente a cada vértice *v*, determina funcionalmente a todos los atributos descendientes de *v*.

Los vértices temporales representan esquemas que tienen como foco de interés la variación de atributos e interrelaciones en función del tiempo. El grafo de atributos será utilizado para construir el esquema de hecho correspondientes al hecho E. Dado un identifier(E) que indica un conjunto de

[1] Por razones de espacio no presentaremos la transformación del modelo de datos al modelo de datos temporal; nos limitaremos a la transformación del modelo de datos temporal al modelo multidimensional temporal.

atributos que identifican a la entidad E, el grafo de atributos (Figura 2) puede ser construido semi automáticamente mediante la aplicación de la siguiente función recursiva modificada de [9]:

```
Function translate (E: Entity): Vertex
{  v = newVertex(E);
   //newVertex(E) crea un nuevo vértice, conteniendo el nombre
   // y el identificador del objeto E
   for each attribute a Î E ½ a Ï identifier(E)  do
   addChild (v, newVertex(a));
   //se agrega un hijo a al vértice v
   for each  entity G connected to E
   by relationship R½card-max(E,R)=1 or R is temporal do
   //se consideran interrelaciones y atributos temporales
           {for each  attribute  b Î R do
           addChild (v, newVertex(b));
           addChild (v, translate(G));
           }
   return(v) }
```

A continuación, ilustraremos con el ejemplo del modelo de datos propuesto (Figura 1, lado derecho), paso por paso, la construcción del grafo de atributos utilizando el algoritmo recursivo. El grafo de atributos resultante de la aplicación del algoritmo se muestra en la figura 2.

```
traslate(VENTA)
      v = <label = VENTA, identifier = {productoID, proveedorID}>
      addchild({productoID, proveedorID}, {cantidad})
      addchild({productoID, proveedorID}, {fecha}); G= PRODUCTO
      addchild({productoID, proveedorID}, traslate(PRODUCTO))
traslate(PRODUCTO)
      v = <label = PRODUCTO, identifier = {productoID}>; G= PRECIO-T
      addchild({productoID}, traslate(PRECIO-T))
      traslate(PRECIO-T)
      v = <label= PRECIO-T, identifier = {proveedorID, tiempo inicial}>
      addchild({proveedorID, tiempo inicial}, {tiempo final}); G = PROVEEDOR
      addchild({productoID, proveedorID}, traslate(PROVEEDOR))
traslate(PROVEEDOR)
      v = <label = PROVEEDOR, identifier = {proveedorID}>; G= LOCALIDAD
      addchild({proveedorID}, traslate(LOCALIDAD))
traslate(LOCALIDAD)
      v = <label = LOCALIDAD, identifier = {localidadID}>; G= LOC-PROV-T
      addchild({localidadID}, traslate(LOC-PROV-T))
traslate(LOC-PROV-T)
      v = <label = LOC-PROV-T, identifier = {localidadID, tiempo inicial}>
      addchild({localidadID, tiempo inicial}, {tiempo inicial)); G = PROVEEDOR
```

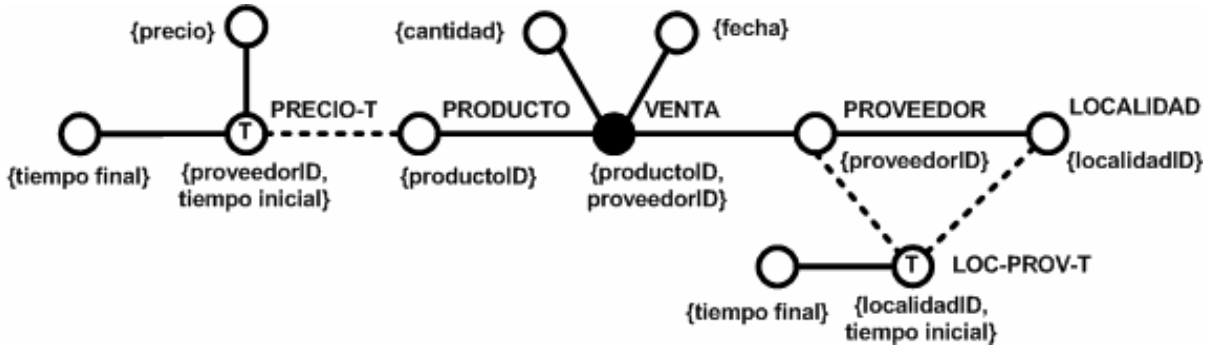


Figura 2. Grafo de Atributos

Cuando ampliamos el modelo entidad interrelación con aspectos temporales, los atributos y las interrelaciones variantes se transformarán en entidades vinculadas con interrelaciones marcadas con T del tipo x-a-muchos (card-max(E, R) > 1), por lo tanto no pueden ser incluidos en la jerarquía para realizar agregaciones. La línea punteada en el grafo de atributos muestra esta particularidad.
Probablemente no todos los atributos representados en el grafo sean de interés para el datawarehouse. Por tal motivo, éste puede ser modificado por el diseñador para eliminar los niveles de detalles innecesarios.

## 2.2 Transformación Informal del Grafo de Atributos al Modelo Multidimensional Temporal

Las dimensiones determinan cómo las instancias de hecho pueden ser agrupadas en forma significativa para el proceso de toma de decisión, éstas deben ser elegidas en el grafo de atributos entre los vértices hijos de la raíz. Las medidas son los atributos vinculados al hecho principal, en el grafo, son todos los vértices asociados a la raíz que no sean identificadores. El último paso en la construcción del esquema de hecho es la definición de las jerarquías en las dimensiones; a lo largo de éstas, los atributos deben ser ordenados en el grafo de forma tal que sean interrelaciones a-uno las que se encuentren entre cada vértice y sus descendientes. La inclusión de atributos e interrelaciones temporales en el grafo (éstos se vinculan mediante líneas punteadas) precisa de una consideración especial en su transformación al esquema de hecho: éstos no formarán parte de la jerarquía para las operaciones de roll-up y drill down, solamente permitirán evaluar cómo atributos e interrelaciones han variado en el tiempo; constituyen, lo que se denomina, jerarquías no estrictas [24].

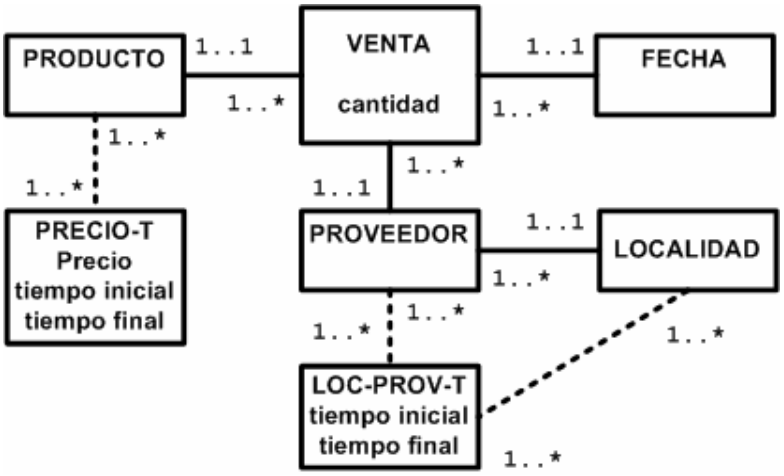


Figura 3. Esquema Multidimensional Temporal

Por lo tanto, el proceso de transformación del grafo de atributos al modelo multidimensional temporal, es decir, la elección de cuales vértices del grafo serán medidas, dimensiones o jerarquías (temporales o no) dependerá de las decisiones del diseñador pero, en general, seguirá el siguiente criterio que utilizaremos en la transformación: la raíz del grafo será el hecho principal; todos los vértices vinculados con la raíz, que no sean identificadores, serán las medidas de hecho; los demás atributos vinculados al hecho serán dimensiones; los vértices vinculados a las dimensiones, que no sean identificadores, serán atributos de la dimensión; los demás atributos serán jerarquías (temporales o no) dentro de las dimensiones; todos los atributos vinculados a una jerarquía, si no son identificadores, serán atributos de éstas, sino serán, también, parte de la jerarquía; todas las jerarquías temporales tendrán asociados un rango temporal. El atributo fecha, asociado al hecho (si lo hubiere) será transformado en dimensión. En la figura 3 se muestra el esquema resultante.

# 3 METAMODELOS Y TRANSFORMACIONES

Un modelo es una descripción de todo o de una parte de un sistema en un lenguaje bien definido; es decir, un lenguaje con una sintaxis y semántica que permita una interpretación automática por parte de herramientas transformadoras de modelos; el proceso de transformación toma uno o más modelos de entrada y produce uno o más modelos de salida. Una regla de transformación de modelos debe definir, evitando cualquier ambigüedad y de forma entendible por los usuarios, la relación implícita que existe entre sus partes. Una forma común de definir reglas es mediante el lenguaje natural; si bien este método puede resultar intuitivo y eficaz para el razonamiento humano, es totalmente ineficaz para el procesamiento automático. MDA no especifica ni prescribe ningún lenguaje para la transformación de modelos, pero actualmente existe un RFP (Request for Proposals) para crear un estándar que permita crear consultas, vistas y transformaciones de modelos en el marco MDA [26]. Las transformaciones, en el contexto de QVT (Query, Views, Transformations), son un supertipo de relación y mapping; las relaciones especifican transformaciones multidireccionales, no permiten crear o modificar modelos, pero sí chequear la consistencia entre dos o más modelos relacionados. El mapping, en cambio, implementa la transformación, es decir, transforma elementos de un dominio en elementos de otro. Por lo tanto, mientras que la relación es una especificación de una transformación que permite chequear la correspondencia entre dos modelos respecto de la relación, un mapping es una implementación que permite, a partir de un modelo de entrada, producir un modelo de salida. Se han propuesto varios lenguajes de transformación: BOTL [15]; ATL [13]; Tefkat [14]; Kent Model [1] y también el uso de sentencias OCL para especificar las transformaciones [6], [7]. Para la especificación de reglas de transformación es esencial el conocimiento de los metamodelos, tanto de los modelos fuente como de los modelos destino [17]. UML [27] es ampliamente recomendado y aceptado como lenguajes de especificación para modelos MDA, aunque no especialmente prescripto.

## 3.1 Metamodelos para las Transformaciones

A continuación, presentaremos los tres metamodelos utilizados para las transformaciones: el metamodelo de datos temporal (Figura 4), el metamodelo de grafos de atributos (Figura 5) y el metamodelo multidimensional temporal (Figura 6).

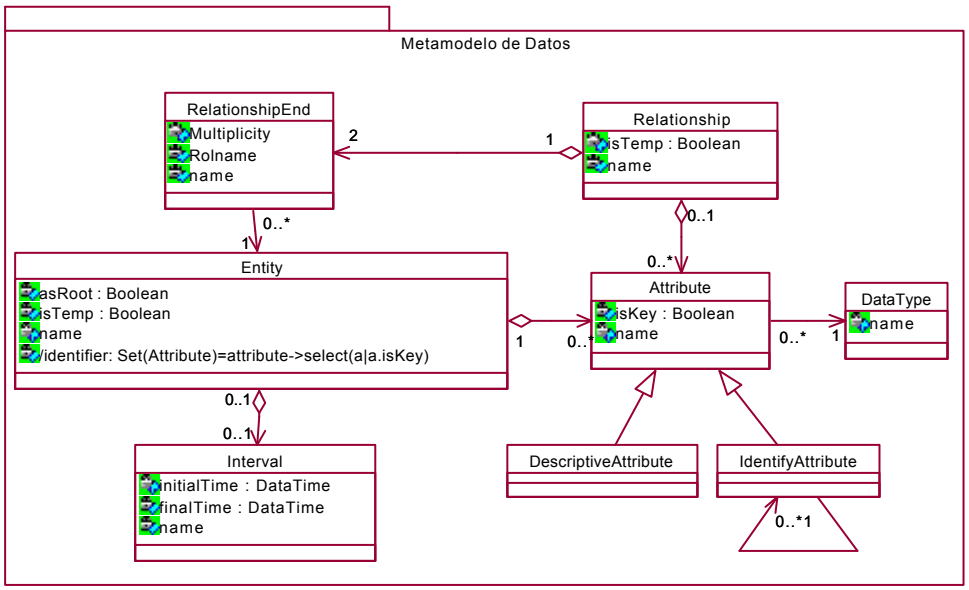


Figura 4. Metamodelo de Datos

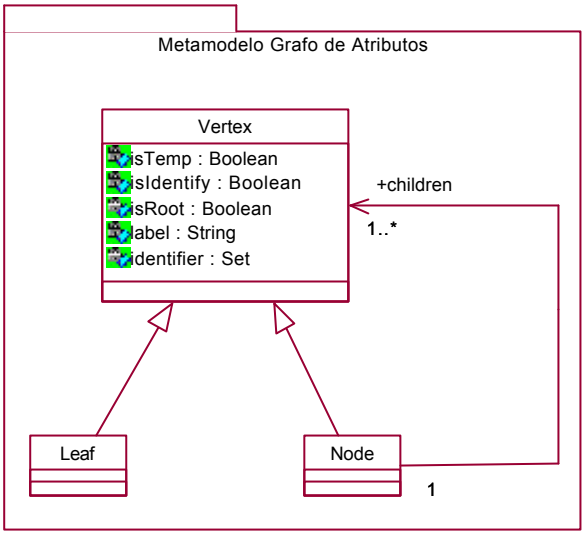


Figura 5. Metamodelo de Grafo de Atributos

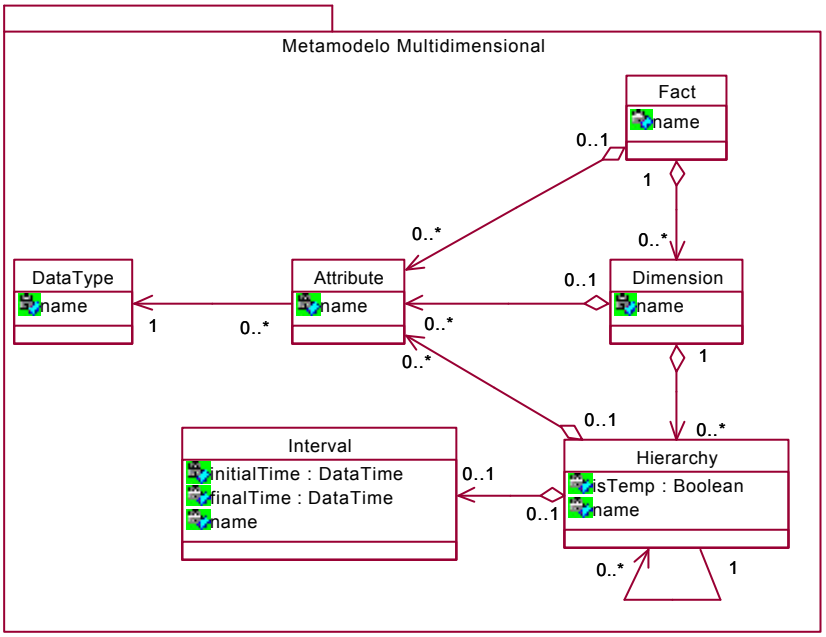


Figura 6. Metamodelo Multidimensional Temporal

## 3.2 Transformación del Modelo de Datos al Grafo de Atributos

En esta sección especificamos formalmente la transformación desde un modelo de datos hacia un grafo de atributos, como paso intermedio en el proceso de transformación hacia el modelo multidimensional temporal, tal como fue descrito en la sección 2. Hemos utilizado el lenguaje declarativo OCL, como alternativa a otros lenguajes específicos para expresar la transformación. OCL es suficientemente expresivo y cuenta con una definición más madura y estable que la de los otros lenguajes mencionados.

```
Context e:Entity :: toVertex(): Vertex
post: e.name = result.label and e.identifier = result.identifier
      -- los atributos de e se convierten en hijos de result
      and e.attribute -> forall(a | e?identifier -> includes(a) or
```

```
      result.children -> includes(newVertex(a))
      -- Se consideran atributos e interrelaciones temporales
      and e.connections -> forAll( Tuple{r, g}|
      (card-max(e, r) = 1 or r.isTemp) implies
      (r.attribute -> forall(b ???result.children -> includes(newVertex(b))
      and result.children -> includes(toVertex(g))
      ))

Definiciones adicionales al metamodelo:

Context Entity :: identifier: Set(Attribute)
-- retorna el conjunto de atributos que son identificadores de la entidad;

Context Entity :: connections: Set(TupleType (r: Relationship, g: Entity))
-- retorna el conjunto de todas las posibles tuplas {r, g} donde r es una
-- interrelación vinculada a self, en tanto que g sea una entidad
-- conectada al extremo opuesto de r;
```

### 3.3 Transformación del Grafo de Atributos al Modelo Multidimensional

Finalmente, la transformación del grafo de atributos al modelo multidimensional es formalmente especificada utilizando expresiones OCL sobre los metamodelos de las figuras 5 y 6, de la siguiente forma:

```
Context v: Vertex :: toFact(): Fact
      Pre: v.isRoot
      -- la raíz del grafo se transforma en  el hecho principal;
      Post: result.name = v.label
      -- todos los vértices vinculados con la raíz, que no sean
      -- identificadores, se transforman en las  medidas del hecho; los
      -- demás atributos  vinculados al hecho, serán dimensiones;
      -- el atributo fecha, asociado al hecho (si lo hubiere),
      -- será transformado en dimension;
      and v.children -> forAll( w | if (w.isIdentify or w.label=”fecha”)
                  then result.dimension->includes (w.toDimension())
                  else result.attribute->includes(w.toAttribute()) endif )

Context v: Vertex :: toAttribute(): Attribute
      Post: result.name = v.label

Context v: Vertex :: toDimension(): Dimension
      Post: result.name = v.label
      -- los vértices vinculados a las dimensiones que no sean
      -- identificadores, serán atributos de la dimensión; los demás
      -- atributos serán jerarquías dentro de las dimensiones;
      and v.children->forAll( w |if h.isIdentify
                  then result.hierarchy -> includes(w.toHierarchy())
                  else result.attribute ->includes(w.toAttribute()) endif)

Context v: Vertex :: toHierarchy(): Hierarchy
      Post: result.name = v.label
      -- las jerarquías temporales tendrán asociados un rango temporal;
      and v.isTemp implies result.isTemp and
                        result.interval -> notEmpty()
      -- todos los atributos vinculados a una jerarquía, si no son
      -- identificadores, serán atributos de éstas, sino serán, también,
      -- parte  de la jerarquía;
      and v.children -> forAll( w |  if a.isIdentify
                  then result.hierarchy -> includes(w.toHierarchy())
                  else result.attribute -> includes(w.toAttribute()) endif)
```

# 4 TRABAJOS RELACIONADOS

Considerando los aspectos temporales en los datawarehouse, se propusieron varias soluciones. En [5] se presentó un esquema estrella temporal que difiere del tradicional en cuanto al tratamiento del tiempo, mientras éste toma al tiempo como una dimensión más, aquel anula la dimensión tiempo y agrega, como atributos de hecho, el tiempo inicial y el final en cada una de las filas de las tablas del esquema. En [24] se describe, entre las características que un modelo de datawarehouse debería tener, la necesidad de considerar los cambios temporales en los datos y las jerarquías no estrictas, compara los modelos existentes contra los requerimientos y observa la falta de soluciones a varios puntos citados. En [18] se presenta el modelo multidimensional temporal y un lenguaje de consulta temporal, donde se agregan marcas de tiempo en las dimensiones o al nivel de instancias (o ambos) para capturar las variaciones en los atributos de las dimensiones.
Entre los trabajos vinculados a la transformación de modelos, en [11] se describe, mediante Meta Object Facility (MOF), la transformación del esquema entidad interrelación al esquema relacional y utiliza sentencias OCL para establecer restricciones en el metamodelo. En [4] se plantean dos fases para la migración de un sistema relacional a un sistema de base de datos OO, en la primera utiliza reglas de transformación para construir un esquema OO que es semánticamente equivalente al esquema relacional, en la segunda fase ese esquema es usado para generar programas que migren los datos relacionales a una base de datos OO. En [10] se estudia la sintaxis y la semántica del modelo entidad interrelación y el modelo de datos relacional y sus transformaciones. En [23] se muestra un marco para representar metadatos acerca de datos fuentes, datos destinos, transformaciones y los procesos y operaciones que crean y administran un datawarehouse. En [2] se estudia el problema de la traducción de esquemas entre diferentes modelos de datos, introducen un formalismo teórico gráfico que permite representar uniformemente esquemas y modelos para comparar diferentes modelos de datos y describir el comportamiento de la traducción. En [21] se establece una conexión formal entre modelos de datos; se utilizan técnicas de metamodelo basado en MOF para representar la transformación, mediante un algoritmo, del esquema entidad interrelación temporal al modelo multidimensional temporal; emplea diagramas de clases MOF y sus correspondientes reglas OCL para establecer restricciones en el modelo y en el metamodelo. En [25] se define una estrategia para verificar formalmente la corrección de transformaciones entre modelos en el contexto de MDE.
Existen trabajos donde, específicamente, se utiliza el enfoque MDA para el diseño de un datawarehouse. En [16] se presenta un método estándar e integrado para el diseño de un datawarehouse; definen el MMD$^2$A (MultiDimensional Model Driven Architecture) como un enfoque para la aplicación del marco MDA en el modelado multidimensional. En [28] se propone un método para el diseño conceptual de un datawarehouse, planteado en tres fases: en la primera se extraen un conjunto de esquemas multidimensionales de las bases de datos operacionales mediante reglas de transformaciones definidas en el marco de MDA, la segunda fase está vinculada con la identificación y la elección de los requisitos del usuario; por último, estos requisitos se usan para seleccionar y refinar los esquemas multidimensionales.

# 5 CONCLUSIÓN Y TRABAJOS FUTUROS

MDA promueve el uso intensivo de modelos en el proceso de desarrollo, se construyen modelos de los sistemas utilizando primitivas de alto nivel de abstracción, luego estos modelos son transformados hasta obtener código fuente del sistema final. Inicialmente, se crea un modelo independiente de la plataforma (PIM); luego, se transforma el modelo anterior a uno o varios modelos específicos de la plataforma (PSM) y, por último, se genera el código a partir de cada PSM. Generar código automáticamente a partir de especificaciones abstractas es una antigua

ambición de las herramientas CASE que, actualmente con MDD (Model Driven Development), ha cobrado nuevo impulso. Si bien las herramientas actuales son capaces de generar clases en algún lenguaje de programación (Java, C#, etc.) y tablas en una base de datos relacional a partir de diagramas de clase, o generar alguna parte dinámica a partir de diagramas de estado, el camino por recorrer todavía es largo.

En el presenta trabajo se presentó una metodología semiautomática para generar un datawarehouse temporal a partir de un modelo de datos temporal; primero se explicó un algoritmo recursivo que permitió armar un grafo de atributos a partir de un modelo de datos y, luego, se estableció informalmente la transformación del árbol de atributos al modelo multidimensional. A continuación, se presentaron los metamodelos del modelo de datos temporales, del grafo de atributos y del modelo multidimensional. Finalmente se presentaron las transformaciones formales utilizando sentencias OCL. En particular, se presentaron transformaciones horizontales, de PIM a PIM.

En trabajos futuros se concluirá con las fases de la propuesta MDA; esto es, se estudiarán las transformaciones del PIM desarrollado a diversos PSM y, luego, su transformación a código fuente.

## REFERENCIAS


[1] Akehurst D.H., Howells W.G.J., McDonald-Maier K.D. Kent Model Transformation Language Proc. Model Transformations in Practice Workshop, part of MoDELS 2005, Montego Bay, Jamaica. 2005

[2] Atzeni P, Torlone R., Schema Translation Between Heterogeneous Data Models in a Lattice Framework. 6th IFIP TC-2 Working Conference on Database Semantics (DS-6), Atlanta, Georgia, May 30-June 2, 1995

[3] Agrawal R, Gupta A, Sarawagi S., Modeling Multidimensional Databases, Research Report, IBM Almaden Research Center, San Jose, California, 1995. ICDE '97.

[4] Behm, A., Geppert, A., Dittrich, K. R. "On the Migration of Relational Schemes and Data Object-Oriented Database System". In Proceedings of Re-Technologies in Information System. Klagenfurt, Austria, Dec 1997.

[5] Bliujute R., Saltenis S., Slivinskas G., and Jensen C. S., Systematic Change Management in Dimensional Data Warehousing. in Proceedings of the Third International Baltic Workshop on Data Bases and Information Systems, Riga, Latvia, 1998.

[6] Cariou, E., Marvie, R., Seinturier, L., & Duchien, L. (2004). OCL for the Specification of Model Transformation Contracts. In J. Bezivin (Eds.), Proceedings of OCL&MDE'2004, OCL and Model Driven Engineering Workshop. Lisbon, Portugal. 2004.

[7] Cariou, E., Marvie, R., Seinturier, L., & Duchien. Model Tranformation Contracts and their Definition in UML and OCL. Technical Report 2004-08, LIFL, April 2004.

[8] Chaudhuri S. and Dayal U., An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD Record 26(1), March 1997.

[9] Golfarelli M., Maio D., Rizzi S., The Dimensional Fact Model: a Conceptual Model for Data Warehouses. International Journal of Cooperative Information Systems, vol 7, n.2&3, 1998.

[10] Gogolla Martin, Lindow Arne, Richters Mark, Ziemann Paul: Metamodel Transformation of Data Models, Workshop in Software Model Engineering, 2002.

[11] Gogolla Martin, Lindow Arne: Transforming Data Models with UML, IOS Press, 2003.

[12] Gupta, H., Harinarayan, V. Rajaraman, A. and J. Ullman. Index Selection for OLAP. Proceeding ICDE 1997.

[13] Jouault, F, Kurtev, I: Transforming Models with ATL. In: Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005, Montego Bay, Jamaica.

[14] Lawley Michael, Steel Jim. Practical Declarative Model Transformation with Tefkat, Lecture Notes in Computer Science, Volume 3844, Jan 2006, Pages 139 – 150.

[15] Marschall Frank, Braun Meter: Model Transformations for the MDA with BOTL In: Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications, CTIT Technical Report TR-CTIT-03-27, Univeristy of Twente, June 2003.

[16] Mazón Jose Norberto, Trujillo Juan, Serrano Manuel, Piattini Mario: Applying MDA to the Development of Data Warehouses. DOLAP 2005: 57-66.

[17] MDA. Model Driven Architecture. 2004. http://www.omg.org/cgi-bin/doc?formal/03-06-01.

[18] Mendelzon A, Vaisman. A Temporal Queries in OLAP. VLDB 2000: 242-253.

[19] Mellor S., Scott K., Uhl A., Weise D. MDA Distilled: Principles of Model-Driven Architecture. Addison-Wesley. 2004.

[20] Neil Carlos, Ale Juan. A Conceptual Design for Temporal Data Warehouse. 31º JAIIO. Santa Fe. Simposio Argentino de Ingeniería de Software. 2002.

[21] Neil Carlos, Pons Claudia. Formalizing the Model Transformation Using Metamodeling Techniques ASSE Argentinean Symposium on Software Engineering. (33 JAIIO04) September 2004. Cordoba. Argentina.

[22] OCL. Object Constraint Language - version 1.5. 2002.

[23] OMG, ed: The Common Warehouse Metamodel Especification. OMG (2000). www.omg.org.

[24] Pedersen T. B., Jensen C. S, Multidimensional Data Modeling for Complex Data. 1998. ICDE 1999:336-345.

[25] Pons C. and Garcia D. "An OCL-based Technique for Specifying and Verifying Refinement-oriented Transformations in MDE". Proceedings MoDELS/UML 2006 “Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genoa, Italy, October 2006" LNCS.

[26] QVT Partners. Initial Submission for MOF 2.0 Query / Views / Transformations RFP. Version 1.0 (2003.03.03).

[27] UML. The Unified Modeling Language Specification – version 1.5. 2003.

[28] Zepeda Leopoldo, Celma Matilde: Aplicando MDA al Diseño Conceptual de Almacenes de Datos. JIISIC 2006: 271-278